TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

Studijní program: N 2612 – Elektrotechnika a informatika

Studijní obor: 1802T007 – Informační technologie

Experimentální systém pro sémantický web

Semantic Web Experimental Environment

Diplomová práce

Autor: **Bc. Václav Materna**

Vedoucí diplomové práce: Mgr. Jiří Vraný

V Liberci 24.4. 2007

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 o právu autorském, zejména § 60 (školní dílo).

Beru na vědomí, že TUL má právo na uzavření licenční smlouvy o užití mé DP a prohlašuji, že **s o u h l a s í m** s případným užitím mé diplomové práce (prodej, zapůjčení apod.).

Jsem si vědom toho, že užít své diplomové práce či poskytnout licenci k jejímu využití mohu jen se souhlasem TUL, která má právo ode mne požadovat přiměřený příspěvek na úhradu nákladů, vynaložených univerzitou na vytvoření díla (až do jejich skutečné výše).

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

Datum:

Podpis:

Abstrakt:

Jádro této práce spočívá ve vytvoření aplikace, umožňující tvořit, editovat a pomocí jednoduchých dotazů také testovat ontologie. Ontologie, jako základní stavební kámen technologie zvané sémantický web, umožňuje určité zachycení reality a sdílení znalostí. Pro tvorbu a reprezentaci ontologií existuje několik různých jazyků s různou „silou“. Většina těchto jazyků má základ v technologii XML. Aplikace popisovaná v této práci, využívá pro reprezentaci ontologií jazyk DAML+OIL. Tento jazyk je podporován organizací W3C a vychází z jazyka RDF, který rozšiřuje. Základní entity jazyka DAML+OIL jsou následující: Třídy – reprezentují určitý obecný objekt (člověk, auto, batoh), vlastnosti – vyjadřují určitou vlastnost objektu. Rozlišujeme objektové vlastnosti a typové vlastnosti (má rodiče, výkon motoru, objem nádrže). Dalším základním prvkem jazyka DAML+OIL jsou instance, neboli individua. Jedná se o konkrétní případy tříd (Petr – instance třídy člověk, BMW – instance auta). Aplikace, kterou prezentuji v této práci, umožňuje tvorbu ontologií v grafickém rozhraní, validaci ontologie (zjištění správnosti jednotlivých hodnot individuí a jejich kardinalit) a také jednoduché dotazování na objektové vlastnosti individuí a tříd. Samozřejmostí je možnost uložení ontologie ve formátu DAML+OIL.

Klíčová slova:

web, ontologie, DAML+OIL, editor ontologií

Abstract:

Core of this work is in creation of the application, which allows building, editing and also testing ontologies, with simple queries. Onthologies, as the base build stone of the semantic web, allows us to capture the reality and sharing knowledge. There are many languages, serving for creation and representation of ontologies. They have various language strenght. Most of these languages have base in the technology called XML. Application described in this work, uses DAML+OIL language for representing ontologies. This language is supported by W3C organization and has base in the RDF language. Basic entities of the DAML+OIL language are as follows: Classes - representing certain object (man, car, bag), properties – expressing certain property of the object. We distinguish object properties and datatype properties (has parents, engine power, tank capacity). The next base element of the DAML+OIL are instance, or individuals. It is concrete instance of classes (Peter – instance of the class man, BMW – instance of the class car). Application, which is presented in this dokument, allows creating ontologies in the graphical interface, validating ontology (check, if values have correct types and cardinality) and also simple questioning on object properties of the individuals and classes. As a matter of course, this application supports saving ontology in the DAML+OIL format.

Keywords:

semantic web, ontology, DAML+OIL, ontology editor

Úvod

Rozvoj webu, jak jej známe dnes započal v první polovině 90. let 20. století. Masivněji se začal rozšiřovat po roce 1995. Od počátku jsou informace na webu uloženy v textové podobě. Jak stránek neustále přibývalo, dostali jsme se až do dnešního stavu, kdy je web neustále se zvětšující haldou stránek, ve které je obtížné nalézt požadované informace. Proto se objevil požadavek, dodat informacím určitý význam – sémantiku. To má být umožněno zaváděním tzv. sémantického webu, o kterém se také někdy mluví jako o webu 2.0.

Základem sémantického webu jsou ontologie. Jsou to právě ontologie, které dodávají informacím význam. Jinými slovy, ontologie umožňují vyjadřovat určité znalosti, které má o informacích autor, ale které se obvykle do klasického textu nedostanou. Tyto znalostní informace mohou však být velmi užitečné například při vyhledávání, nebo při strojovém zpracování textu.

Ontologie jsou vytvářeny pomocí speciálních ontologických jazyků. Těchto jazyků je několik a liší se svou „vyjadřovací silou“, tedy tím kolik obsahují elementárních objektů. Se vzrůstající vyjadřovací silou sice roste množství informací které jsme schopni ontologií reprezentovat, ale na druhé straně se zvyšuje pracnost tvorby takové ontologie, roste pravděpodobnost chyby a zvyšují se nároky na aplikace sloužící pro reprezentaci ontologie. I zde tedy platí heslo, že méně je někdy více.

Jak již bylo uvedeno, jsou ontologie reprezentovány pomocí speciálního jazyka. Ovšem tvorba ontologie „v poznámkovém bloku“ přímo v daném jazyce je krajně nepohodlná a neefektivní. Proto se objevují aplikace, které umožňují ontologie tvořit, editovat a ukládat. Takováto aplikace, je i předmětem této práce.

Aplikace umožní načítání, editování a ukládání ontologie vytvořené v jazyce DAML+OIL. Umožní ontologii validovat a zobrazí případné chyby uživateli. Bude také možné ontologii testovat pomocí dotazů.

1. Sémantický web

1.1 Co je to sémantický web

Myšlenka sémantického webu, byla poprvé představena v roce 2001. Dá se říci, že sémantický web, je rozšířením současného webu. Ten byl od začátku koncipován tak, že data v něm uložená jsou srozumitelná pro člověka. Ale chceme – li v těchto datech vyhledávat, nebo je jinak strojově zpracovávat, je tato reprezentace nevhodná. Sémantický web vnáší strukturu do informací uložených na současném webu. Umožňuje strojově vykonávat různé důmyslné úlohy. Zajímavý příklad využití sémantického webu, který zároveň demonstruje jeho potenciál, dává ve svém článku [1] Tim Berners-Lee, jeden z tvůrců současného webu a ředitel konsorcia W3C: „Zábavní systém zrovna vyřvává „We Can Work It Out“ od Beatles, když zazvoní telefon. Jakmile ho Pete zvedne, telefon ztlumí zvuk posláním zprávy všem ostatním *lokálním* zařízením, která mají *ovládání hlasitosti*. Petova sestra Lucy, volala z doktorovy kanceláře: „Matka potřebuje konzultovat specialistu a pak musí podstoupit sérii fyzioterapeutických sezení. Dvakrát týdně, nebo tak. Nechám svého „agenta“ zařídit schůzky.“ Pete okamžitě souhlasí s tím, že pomůže s odvozem. V doktorově kanceláři Lucy instruuje svůj přenosný prohlížeč, aby z doktorova „agenta“ získal informace o matčině *předepsané léčbě*, prohledal několik seznamů *poskytovatelů* a hledal ty, kteří *vyhovují* matčině pojištění, v *okruhu 20 mil* od jejího *domu* a s *hodnocením výborný*, nebo *velmi dobrý* na důvěryhodných ratingových službách. Potom začne hledat shodu mezi volnými *časy schůzek* vybraných terapeutů a nabitými rozvrhy Peta a Lucy. V několika minutách „agent“ nabídne plán. Petovi se nelíbí – Univerzitní Nemocnice je na

druhém konci města a na cestě zpět by musel jet v dopravní špičce. Nastavil svého vlastního agenta, aby zopakoval hledání se striktněji zadanými omezeními ohledně *místa* a *času.* Luciin agent *plně důvěřuje* Petovu agentu v kontextu současné úlohy, proto automaticky asistoval dodáním informací, které již získal. Nový plán je prezentován téměř okamžitě: o mnoho bližší klinika a dřívější čas, ale jsou zde dvě varování. První, Pete bude muset přesunout pár svých *méně důležitých* schůzek. Druhé bylo něco o seznamu pojišťovací společnosti, který odmítl zařadit tohoto *poskytovatele* mezi *fyzioterapeuty*. S dovětkem, že typ služby a pojišťovací plán byly bezpečně ověřeny jinými způsoby a možností zobrazit detaily. " Text kurzívou znamená termíny, jejichž sémantika – význam byl definován skrze sémantický web.

Uvedený příběh by se s možnostmi současného webu nemohl stát. Vyhledávání informací dnes funguje pomocí klíčových slov a i když jsou některé vyhledávací algoritmy na vysoké úrovni, nemohou možnostem sémantického webu konkurovat.

Aby sémantický web fungoval, je potřeba mít přístup ke kolekcím informací a sadě odvozovacích pravidel, které se použijí při automatickém usuzování. Obvyklé systémy reprezentace znalostí jsou centralizované. Všichni mají přístup ke stejným definicím základních elementů (člověk, auto, …), ale centralizace nejde dohromady s myšlenkou webu. Navíc množství informací v takovém centrálním systému by bylo obrovské a nezvládnutelné.

Současný sémantický web jde cestou menších a decentralizovaných systémů.

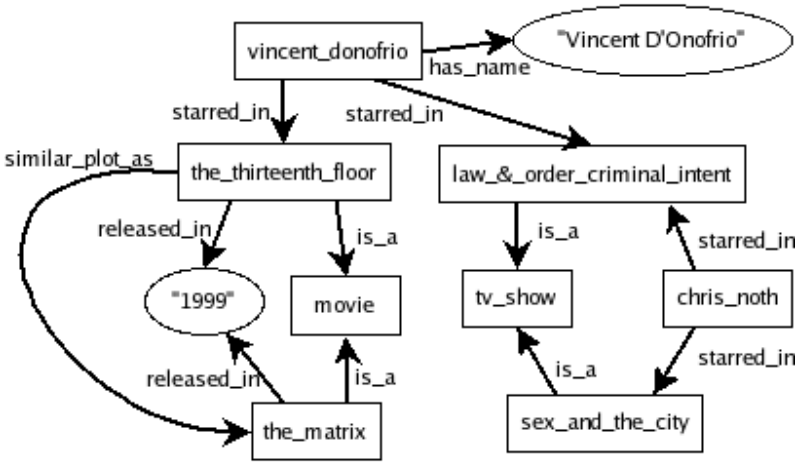
1.2 Technologie využité v sémantickém webu

Jak již možná vyplynulo z předcházejícího textu, sémantický web je souborem několika technologií. První z technologií, které sémantický web využívá je XML (eXtensible Markup Language). XML je jazyk využívající tzv. tagy, neboli značky. Tyto tagy jsou párové a dávají určitý význam objektům vloženým mezi ně. Tagy lze pochopitelně vkládat a vytvářet tak hierarchickou strukturu. Uživatel může vytvářet vlastní tagy a dávat tak dokumentu libovolnou strukturu, ale neříká tím nic o tom, co tato struktura znamená. Význam je vyjádřen pomocí RDF.

RDF (Resource Description Framework) je jazyk prosazovaný skupinou W3C, který umožňuje reprezentovat znalosti. Jak jsme již řekli, na webu je uloženo hodně informací, ale tyto informace jsou pro počítač „nepochopitelné“. Jediné informace, kterým počítač zobrazující stránku rozumí, jsou informace o formátu. Ví kde leží obrázky, jakou velikost má písmo, ale nedokáže nijak zpracovávat informace obsažené v textu. O co nám vlastně jde? Ne o to, aby počítač rozuměl informacím uloženým na webu, ale o to aby na základě určitých logických souvislostí, dokázal s daty automaticky manipulovat k našemu užitku. Celý tento požadavek si lze představit jako web složený z databází. Jedna databáze nabízí seznam produktů i s jejich popisem, druhá pak hodnocení produktů. Databáze prodejce potom poskytuje databázi produktů, které jsou na prodej. V současnosti si musí určitý zájemce o koupi produktu všechny informace „pospojovat“ sám. Musí si najít specifikaci produktu, musí si vyhledat jeho recenze a potom ještě prodejce, který daný produkt prodává a má ho na skladě. Je pravda, že je možné i bez prostředků sémantického webu napsat aplikaci, která si tyto informace spojí. Nicméně kdyby existoval standard, který by tuto funkcionalitu umožňoval, podnítilo by

to tvorbu potřebných aplikací a také by se usnadnila práce vývojářů. Jak uvádí J. Tauberer v [2], je zde několik bodů, které by měl tento standard respektovat:

- Soubory na sémantickém webu, musí být schopny vyjádřit informaci flexibilně. Realitu nelze zachytit v tabulkách, relačních databázích nebo hierarchiích jako je XML. Jako příklad uvedu obrázek publikovaný na webu www.xml.com. Zachycuje informace o filmech a televizních seriálech.



Obrázek 1: Znalost jako graf

- Chceme, aby soubory na sémantickém webu byly schopny odkazovat se jeden na druhý. Aby soubor o cenách produktů dodaný dodavatelem a soubor recenzí dodaný nezávisle spotřebitelem, měl možnost vyjádřit, že mluví o stejném produktu. Prosté použití jména produktu nestačí, protože na světě se mohou vyskytovat dva produkty s totožným názvem a v sémantickém webu nechceme nejednoznačnost, aby mohl počítač zpracovat informace s jistotou. Sémantický web

potřebuje globálně unikátní identifikátory, které mohou být přiřazovány decentralizovaným způsobem. Jak již bylo zmíněno výše, centrální správa znalostí je pro myšlenku sémantického webu nevhodná.

- Pro vytváření tvrzení o věcech, se používají slovníky. Tyto slovníky musí být schopny být „smíchány“ dohromady. Například slovník o televizních seriálech vytvořený seriálovými fanoušky a slovník o filmech vytvořený příznivci filmu, musí být schopny být použity dohromady ve stejném souboru, který se týká například herců (a vyjádřit, že herec hrál současně ve filmu i seriálu).

Všechny tyto požadavky splňuje právě jazyk RDF. Je to svým způsobem druh datového modelu. Zde je příklad vyjádření několika informací z obrázku 1, pomocí RDF/XML:

```
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ex="http://www.example.org/">
    <rdf:Description
rdf:about="http://www.example.org/vincent_donofrio">
        <ex:starred_in>
            <ex:tv_show
rdf:about="http://www.example.org/law_and_order_ci" />
        </ex:starred_in>
    </rdf:Description>
    <rdf:Description
rdf:about="http://www.example.org/the_thirteenth_floor">
        <ex:similar_plot_as
rdf:resource="http://www.example.org/the_matrix" />
    </rdf:Description>
</rdf:RDF>
```

RDF byl vytvořen roku 1999 jako standard rozšiřující XML. Jazyk RDF umožňuje rozložení jakékoliv znalosti na malé části. Tyto části se nazývají

trojice a jsou to: zdroj (předmět, entita), vlastnost (predikát) a hodnota vlastnosti (objekt). Tyto trojice se také nazývají tvrzení.

- Zdroj – můžeme se na něj odkazovat pomocí URI[1]. Zdroj nemusí být pouze webová stránka, URI identifikátor může být přiřazen jakémukoliv objektu, abstraktnímu pojmu, osobě atp. Při vytváření URI se uplatňují tzv. jmenné prostory. Jediný požadavek je, aby URI bylo globálně unikátní.
- Vlastnost – definuje binární relaci mezi zdrojem a atomickými hodnotami. I predikát lze zadávat pomocí URI
- Objekt – specifikuje hodnotu vlastnosti pro nějaký předmět. Objektem je buďto zdroj a nebo literál.

Koncept trojic a použití URI identifikátorů činí RDF unikátním a silným prostředkem. Splňuje tak všechny požadavky na decentralizované distribuování znalostí.

1.3 Ontologie

Ontologie tvoří základní stavební kámen sémantického webu. Tento pojem pochází původně z filozofie 18. století. Ontologie je filozofy chápána jako věda o bytí a jsoucnu. Oproti tomu ontologie v informatice se snaží o zachycení podstaty reálného světa, tak aby to bylo využitelné v informačních technologiích. Tato snaha začala v 70. letech 20. století a později se rozdělila na 2 proudy. První proud se dodnes projevuje při návrhu software pomocí objektových modelů. Je to tedy základ objektové analýzy, objektově orientovaného programování a objektových databází.

[1] Uniform Resource Identifier – jednoznačně identifikuje prostředek na globální úrovni

Druhý proud do praktického využití tak rychle nepronikl a je stále spíše doménou akademického výzkumu. Tento směr, založený na logických kalkulech, byl využíván zejména v aplikacích umělé inteligence. Postupně se zájem posouval směrem ke sdílení znalostí různými institucemi. Vzniká snaha o vytvoření pojmového aparátu, pomocí nějž by bylo možné sdílet znalosti z několika zdrojů. Dochází k zavedení pojmu ontologie do informatiky. Představuje sémantický systém a určitou množinu základních pojmů, s jejichž pomocí je možná výměna znalostí a jejich strojové využití bez nutnosti znalosti jejich struktury.

Jeden z duchovních otců ontologií, T. Gruber ji definoval jako „explicitní specifikaci konceptualizace“. Tato definice byla rozšířena W. Borstenem, podle nějž je ontologie „formální specifikace sdílené konceptualizace“. Jinými slovy, ontologie je systémem zachycení reality, který je přenosný a je možné jej sdílet. Pokud by ontologie kteroukoliv z těchto dvou vlastností postrádaly, byly by pro sémantický web nepoužitelné.

Nejtypičtější případ ontologie na webu má taxonomii[2] a sadu odvozovacích pravidel. Taxonomie definuje třídy objektů a vztahy mezi nimi. Třídy odpovídají pojmům, jako učitelé, studenti, předměty a přednášky. Například adresa může být definována jako typ polohy a směrovací číslo může být definováno tak, že lze aplikovat pouze na objekty typu poloha. Třídy, podtřídy a relace mezi entitami jsou velmi silným nástrojem. Odvozovací pravidla dodávají ontologiím další potenciál. Ontologie jsou v současném sémantickém webu reprezentovány určitým jazykem, který z pravidla vychází z jazyka RDF. O ontologických jazycích se zmíním podrobněji v další části práce.

[2] Taxonomie – pochází z řeckých slov taxis – pořádek, řád a nomos – právo, věda. Taxonomie jsou tvořeny taxonomickými jednotkami, taxony, které tvoří hierarchickou strukturu a obvykle obsahují relace dítě - rodič

V praxi se používají dva způsoby vytváření ontologií: shora-dolů a zdola-nahoru. V prvním případě, se začíná od obecných pojmů a postupuje se ke speciálním. Pro sémantický web je typičtější druhý případ.

Úsilím ontologického inženýrství je tedy tvorba formálního systému k zachycení reality a také logické odvozování v rámci této oblasti, modelování a dotazování. V praxi se setkáme s řadou typů ontologií, zpracovávajících tyto schopnosti v různém stupni. Nejjednodušší jsou terminologické, či lexikální ontologie. Základem ontologií tohoto typu jsou termíny. Tyto termíny nejsou dále formálně definovány. Umožňují vyjádřit vztah ekvivalence či podobnosti mezi termíny, případně vztah celek – část.

Pokročilejší ontologické modely dokáží zaznamenat větší množství vztahů a modelovat tak složitější domény. To si však vyžaduje složitější ontologický jazyk. O jednotlivých ontologických jazycích se zmíním později.

Jak probíhá tvorba ontologie? Máme určitou problémovou oblast – doménu, kterou si přejeme analyzovat. Nejprve je nutné získat její pojmový aparát určující entity, vztahy mezi nimi atp. Teprve pokud máme tyto prvky k dispozici, jsme schopni entity skládat do určitých modelových situací a odvozovat nějaké závěry. Zde můžeme vidět dvě základní části ontologií. Jmenný systém identifikující entity a základní sadu znalostí vztahů. Pokud některý z těchto prvků chybí, ontologie ztrácí smysl. První ze základních účelů ontologií, tedy analýza problémové domény, je jasný. Další stejně důležitou vlastností ontologií, je sdílení znalostí. Ontologie mají poskytovat základní rámec k zachycení reality bez předchozích znalostí struktury těchto informací. Vytvořit takový systém zachycení reality však není jednoduché, protože můžeme modelovat jen takové objekty, které lze vyjádřit pomocí prostředků základního rámce. Tedy jazyka použitého pro modelování ontologie. Jednodušším jazykem tak postihneme jednodušší vztahy a složitější pak

musíme skládat. Benefitem je na druhé straně jednodušší zpracování takové ontologie. Naopak složitější jazyky umožňují vyjádřit složitější vztahy, ale na druhou stranu je třeba větší programový aparát při zpracování.

1.3.1 Struktura ontologie

Struktura ontologie je bez ohledu na použitý jazyk podobná. Jak popisuje ve své práci V. Svátek [3], ontologie se skládají z následujících částí:

- *Třídy, koncepty, kategorie, rámce* – Element třídy zajišťuje pojmenování a identifikaci, označuje množiny objektů. Na rozdíl od objektově orientovaného programování, třídy neobsahují metody, ale jsou definovány spíše jako unární relace. Pro třídu lze použít podmínky postačitelnosti, nebo nutnosti. Mezi třídami vždy existuje hierarchie, přičemž je obvykle aplikována vícenásobná dědičnost.
- *Individua, instance* – Instance je konkrétní případ nějaké třídy (např. Petr je instancí třídy člověk). Zajímavostí je, že instance může provizorně fungovat i bez třídy. Ontologie z principu pracují s abstraktními typy objektů, proto některé z ontologií instance interně nepodporují.
- *Relace, funkce, sloty, vlastnosti, role, atributy* – I zde je možné definovat relace mezi třídami, podle složitosti ontologického jazyka lze vytvářet složitější, nebo jednodušší podmínky. V případě ontologií nejsou relace podřízeny příslušným třídám, ale vystupují jako samostatné objekty. Mohou být i specifickým typem třídy.
- *Primitivní hodnoty, datové typy* – Relace může být také elementární primitivní hodnota. Tento vztah třída – hodnota je v některých

případech nahrazen vztahem třída – instance. Datový typ vystupuje jako třída a jeho konkrétní hodnota jako instance (4 je instance třídy integer). Odlišností proti objektovému programování je fakt, že sloty mohou nabývat více hodnot současně, což je v objektovém programování nepřípustné.

- *Axiomy, pravidla* – Vedle příslušnosti ke třídám a relacím lze v ontologiích vytvářet další logické formule vyjadřující ekvivalenci, subsumpci či disjunktnost tříd.

1.3.2 Ontologické jazyky

Jak již bylo uvedeno, pro tvorbu a reprezentaci ontologií se používají speciální jazyky. Těchto jazyků se za dobu existence ontologického inženýrství vyvinulo několik desítek. Čerpat budu z práce V. Svátka [3].

Cyc

Jeden z prvních projektů pokoušejících se o zachycení znalostí o světě je projekt Cyc (ze slova enCYClopedia). Projekt byl zahájen v roce 1984 pod vedením D. Lenata. Cílem bylo shromáždit všeobecné znalosti, které by v expertních systémech fungovaly po boku znalostí expertních a zabraňovaly absurdním situacím (doporučení testu gravidity pro mužského pacienta). Ačkoliv projekt patří do oblasti ontologií, autoři mluví spíše o mikroteoriích složených z dílčích tvrzení. Pro formální reprezentaci se využívá vlastního jazyka CycL a používá notaci převzatou z jazyka LISP. Dlouhou dobu byl tento projekt orientován na komerční aplikace, teprve nedávno v roce 2001 se přiklonil k zásadě veřejné přístupnosti zdrojů.

Ontolingua

Vzniká na začátku 90. let pod vedením T. Grubera ze stanfordské Knowledge System Laboratory. Jazyk je od začátku koncipován jako otevřený. Jeho cílem je vyvinout dostatečně mocný a přehledný jazyk, který umožní sdílení ontologií v rámci odborných komunit používajících vzájemně nekompatibilní znalostní systémy. Jazyk je koncipován jako nadstavba jazyka KIF (Knowledge Interchange Format), který využívá syntaxe jazyka LISP. Ontolingua byla od začátku koncipována jako „mezi jazyk“, určený primárně k výměně ontologických informací mezi systémy, které interně používají vlastní reprezentaci. Z toho plynou omezené možnosti odvozování ontologií přímo v tomto jazyce, na druhou stranu však Ontolingua vzhledem ke své rozšířenosti často plnila úlohu jazyka první volby pro tvorbu ontologií nezávisle na konkrétním znalostním systému.

OCML

Vznik jazyka OCML (Operational Conceptual Modelling Language) byl motivován omezenými možnostmi jazyka Ontolingua. Za jeho vznikem stojí E. Mott z Open University ve Velké Británii. Cílem bylo navrhnout jazyk, který by výrazněji podporoval přímý vývoj programových aplikací, bez nutnosti model překládat do jiného jazyka. Vývoj OCML byl proto těsně propojen s tvorbou jeho interpretu, implementovaného v prostředí CommonLISP. Základem tohoto interpretu jsou algoritmy pro prologovské dokazování a dědění v hierarchii tříd. Třídy a jejich atributy jsou ovšem chápány jako unární, respektive binární relace, takže primární vnitřní reprezentací jsou Hornovy klauzule[3]. Deklarativní část jazyka OCML je prakticky shodná s

[3] Hornovy klauzule – mají stěžejní roli v logickém programování. Klauzule obsahuje právě jeden pozitivní literál a několik (nejméně jeden) negativních.. Vyjadřuje implikaci.

jazykem Ontolingua. Vedle toho je však podporována řada konstruktů převzatá z procedurálních jazyků expertních systémů (cykly, podmínky, produkční pravidla). V OCML je proto možné napsat i aplikaci, která nemá s ontologiemi nic společného. Tento jazyk se však, z části i vinou vazby na LISP jako implementační prostředí, příliš nerozšířil mimo Open University.

OKBC a XOL

Roku 1993 začal vznikat návrh aplikačního rozhraní, které by umožňovalo otevřenou komunikaci mezi znalostními systémy, OKBC (Open Knowledge Base Connectivity). Protokol specifikuje jak způsob předávání konstruktů znalostních bází (třídy, individua, sloty), tak i volání operací nad těmito konstrukty. Soubor konstruktů podporovaných OKBC do značné míry odpovídá souboru konstruktů ontologických jazyků jako je Ontolingua. Proto se jednoduchá verze OKBC – OKBC Lite, stala v roce 1999 výchozím bodem pro návrh dalšího ontologického jazyka, XOL (eXtensible Ontology Language). Motivací jeho rozvoje byla potřeba sdílet struktury znalostí o genovém výzkumu, které žádný z do té doby existujících existující systémů nevyhovoval. Podstatnou inovací proti předchůdcům bylo zakotvení v syntaxi XML, což umožnilo využití už existujících nástrojů pro tento jazyk.

SHOE

SHOE (Simple HTML Ontology Extension) je prvním jazykem vzniklým specificky pro účely přidání sémantiky k webovým stránkám. Byl vyvinutý roku 1996 týmem J. Hendlera na University of Maryland. SHOE umožňuje začleňovat do zdrojového kódu webových stránek metadata o objektech kterých se tyto stránky týkají, a také samotné ontologie, které definují sémantiku těchto metadat. Proti jazykům typu Ontolingua je SHOE podstatně

jednodušší. Zachycuje pouze třídy a relace bez odlišení facet (slotů). Oproti budoucím webovým jazykům, je zde možné definovat relace s libovolnou aritou. V jazyce SHOE se instance implicitně ztotožňují s fyzickými stránkami, nebo jejich částmi. Tím pádem webový objekt (stránka identifikovaná URL) do jisté míry splývá s objektem reálného světa. To je v pozdějších jazycích eliminováno zobecněním pojmu zdroje (viz. RDF), který už nemusí být fyzická stránka HTML.

Ontobroker

Vzniká ve stejné době jako SHOE, na univerzitě v Karlsruhe. Idea projektu je stejná se SHOE – obohacování webových stránek o sémantické anotace. Rozdíl je však v architektuře, která je v jazyce Ontobroker narozdíl od SHOE důsledně centralizovaná. Předpokládá se existence ontologického serveru, který spravuje ontologie a editace je umožněna pouze oprávněným a kvalifikovaným uživatelům. Také sbírání dat z anotovaných stránek neprobíhá náhodně, ale jedná se o cílené návštěvy stránek registrovaných poskytovatelů informací, patřící k určité profesní komunitě. V důsledku toho nepoužívá Ontobroker jednotný jazyk pro ontologie a anotace. Jazyk anotací je jednoduchým obohacením HTML o dodatečné atributy (na rozdíl od SHOE tak zavádí vlastní elementy). Jazyk ontologií je propracovaný formalismus tzv. F-logic, jednoho z řady logických kalkulů vytvořených pro potřeby rámcových systémů.

RDFS

Koncem 90. let bylo zřejmé, že nutnou podmínkou rozšíření sémantických metadat je jejich kompatibilita s metadatovým standardem vytvořeným konsorciem W3C, tj. s RDF (viz. výše). Postupně se prosadil

názor, že grafový model RDF je na rozdíl od stromového modelu XML dostatečně otevřený pro vyjádření sémantiky metadat prostřednictvím ontologií. Příklon k RDF se časově shoduje s vytvořením iniciativy označované jako sémantický web. Problematika ontologických jazyků je od té doby podřízena právě sémantickému webu.

RDFS (RDF Schema) je první z ontologických jazyků, orientovaných na RDF. Vznikl v roce 1999 přímo pod vedením W3C. Tento jazyk představuje nadstavbu, která doplnila do struktury RDF třídy, sloty s možností stanovit definiční obor a obor hodnot, nad třídami i sloty je možné definovat hierarchii. Oproti tradičním ontologickým jazykům chybí RDF možnost precizněji specifikovat podmínky příslušnosti ke třídám a zcela postrádá datové typy.

DAML+OIL

V polovině roku 2000 byl zahájen projekt DAML (DARPA Agent Markup Language), který sponzorovala vojenská organizace DARPA. Na tomto projektu se od počátku podílela řada významných jmen z oblasti výzkumu sémantického webu, včetně Tima B. Lee. Cílem bylo vytvořit jazyk pro RDF, s větší vyjadřovací silou než má RDFS. Jazyk byl nazván DAML-ONT. Koncem 90. let dospěla evropská i americká větev výzkumu webových ontologií k potřebě postavit nové jazyky, které by umožňovaly konstrukci složitějších podmínek, při zachování výhodných vlastností pro výpočty. Volba padla na deskripční logiku[4], která se stala základem jazyka OIL (Ontology Inference Layer). Sloučením jazyka DAML-ONT a OIL vznikl jazyk DAML+OIL. Jeho základem jsou třídy, které lze reprezentovat buď svým

[4] Deskripční logika – tento pojem zastřešuje řadu formalismů. Jejich společným rysem je snaha zachytit strukturu tříd a relací. Od tradičních ontologických jazyků je odlišuje to, že nepředpokládají apriorní vymezení vztahu třídy a podtřídy (taxonomie), ale tvoří je dynamicky na základě vyhodnocování jejich popisů.

jménem tj. URI (pojmenované třídy), nebo logickým výrazem (anonymní třídy). Pro tvorbu tříd se používají konstruktory uvedené v následující tabulce.

intersectionOf unionOf complementOf	Booleovský výraz nad třídami
oneOf	Třída jako množina primitivních hodnot
toClass hasClass	Třída splňující univerzální, resp. Existenční omezení na slot (tj. na třídy které jsou jeho hodnotami)
hasValue	Třída splňující omezení na konkrétní hodnotu slotu
minCardinalityQ maxCardinalityQ CardinalityQ	Třída splňující omezení na kardinalitu slotu

Tabulka1: Konstruktory pro tvorbu tříd

Konstruktory je možné libovolně skládat a vytvářet tak složité výrazy. Vlastní náplň ontologie tvoří axiomy, vybudované nad výrazy reprezentující třídy. Typy axiomů jsou uvedené v následující tabulce.

subClassOf	vztah obecnější a speciálnější třídy
sameClassAs	ekvivalence tříd
disjointWith	prázdný průnik tříd
subPropertyOf	vztah obecnější a speciálnější vlastnosti (slotu)
samePropertyAs	ekvivalentní vlastnosti
inverseOf	inverzní vlastnosti
transitiveProperty	tranzitivita vlastností
uniqueProperty	unikátní vlastnost
unambiguousProperty	inverzní vlastnost k dané vlastnosti je unikátní
sameIndividualAs	totožné individuum
differentIndividualFrom	odlišná individua

Tabulka 2: Typy axiomů

Jazyk DAML+OIL tvoří tzv. vrstvená architektura. Znamená to, že jazyk obsahuje XML elementy pocházející z různých jazykových prostorů. Konkrétně DAML+OIL obsahuje elementy z prostorů RDF, RDFS a DAML. Tato architektura funguje tak, že jazyk z vyšší vrstvy dědí elementy z jazyka nižší vrstvy. To zaručuje zpětnou kompatibilitu a agent napsaný pro jazyk

RDFS bude schopen provést základní syntaktickou analýzu. Nedostupné pro něj budou jen informace z vyšší úrovně reprezentované zde jmenným prostorem daml.

OWL

Na základě zkušeností s jazykem DAML+OIL vznikl pod vedením skupiny W3C – Ontology Working Group nový ontologický jazyk OWL. Jeho specifikem je vyčlenění minimální množiny konstruktů jazyka a jejich označení jako OWL Lite. To usnadní vývoj nástrojů pro práci s tímto jazykem, protože jejich vývoj byl pro jazyk DAML+OIL a plnou verzi jazyka OWL velmi komplikovaný. OWL Lite podporuje jen některé typy lokálních omezení na sloty, jsou oslabeny možnosti využití anonymních tříd a omezení na kardinalitu. V současné době existují tři verze jazyka OWL. OWL Lite, OWL DL a OWL Full.

1.3.3 Agenti sémantického webu

Jak už bylo zmíněno v některých předchozích odstavcích, existuje cosi jako agenti sémantického webu. Pravá síla sémantického webu totiž vynikne ve chvíli, až bude vytvořeno více programů, které budou schopné vybírat obsah webu z různých zdrojů, zpracovávat tyto informace a sdílet je s jinými programy. Tyto programy se nazývají agenti. Jejich efektivita poroste s množstvím dostupného strojově čitelného webového obsahu. Dokonce i agenti, kteří nebyli navrženi pro práci s jinými agenty, budou moct s nimi sdílet informace, protože data již budou mít potřebnou sémantiku, danou standardním jazykem. Mnoho automatizovaných webových služeb existuje už dnes bez

sémantiky. Omezujícím faktem je to, že je není možné využívat pomocí softwaru, který nebyl napsán přímo pro ně.

Je možné, že dalším krokem, by se mohl sémantický web dostat i do našeho reálného prostoru, a rozšířit se do fyzického světa. Jak jsem již uvedl URI může odkazovat na jakýkoliv prostředek, to znamená i na fyzické entity. To znamená, že pomocí jazyka RDF můžeme popsat zařízení jako mobilní telefon, nebo televize. Tyto zařízení můžou pak „inzerovat“ svojí funkcionalitu – jaké mají funkce a jaké ovládací rozhraní. Tím bude umožněna jejich vzájemná interakce. Třeba že zvednutý telefon ztiší okolní zařízení mající vlastnost ovládání hlasitosti.

2. Aplikace

Cílem této práce je vytvoření aplikace, která by sloužila jako testovací prostředí pro experimenty se sémantickým webem. Následující kapitoly budou věnovány popisu vytvořené aplikace a jejím schopnostem.

2.1 Použité technologie

Aplikace podporuje ontologie napsané v jazyce DAML+OIL. Při volbě ontologického jazyka jsem chtěl zvolit některý s velkou vyjadřovací silou. V úvahu tak přicházely pouze DAML+OIL a OWL. Konečný výběr byl ovlivněn faktem, že k jazyku DAML+OIL jsem měl lepší dokumentaci a k dispozici byly i poměrně propracované testovací ontologie.

Pro vývoj aplikace samotné jsem se rozhodl pro jazyk C# ve verzi 2.0. Výběr byl dán tím, že v jazyce C# pracuji a nebylo tak nutné studovat od základů jiný programovací jazyk. Přínosem C# je rovněž kvalitní soubor dodávaných knihoven, které mají mimo jiné propracovanou podporu práce s XML dokumenty. Nebylo tak nutné zabývat se parsováním samotné XML struktury, ale mohl jsem se plně soustředit na práci s DAML+OIL strukturou. Program byl psán v prostředí Microsoft Visual C# 2005 Express Edition, což je odlehčená verze Visual Studia určená pouze pro C# a je dostupná zdarma i ke komerčnímu použití. Aplikace je ke stažení na stránkách firmy Microsoft.

K interní reprezentace ontologie po načtení z DAML+OIL jsem se rozhodl použít DataSet. DataSet, je jeden ze základních stavebních kamenů

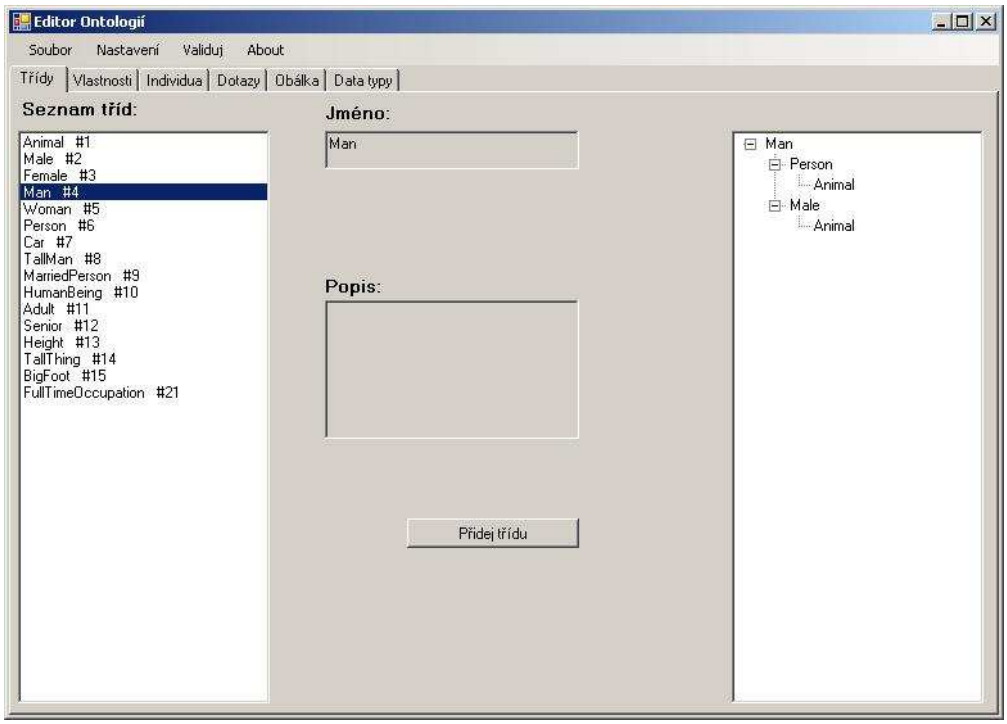
ADO.NET. ADO.NET je technologie, konkrétně kolekce tříd, která je součástí MS .NET frameworku. Její primární účel, je zprostředkovat přístup ka datům z různých zdrojů, jako je SQL server, nebo XML. ADO.NET je na technologii XML postavená a proto umožňuje komunikovat i s technologiemi, které o ADO.NET ani nevědí. DataSet je tedy v podstatě reprezentace dat uložených interně v XML. DataSet je v podstatě cache uložená v paměti a obsahující data. Skládá se z kolekce tabulek a umožňuje specifikovat relace mezi nimi. Umožňuje rovněž určovat omezení pro sloupce tabulek a jednoduše řečeno, umožňuje k uloženým datům přistupovat jako by byly uloženy v klasické relační databázi.

Reprezentace ontologie formou DataSetu přinesla několik podstatných výhod. Na rozdíl od nativní reprezentace v XML, jsou mezi daty určeny vazby. Tak například individuum má vazbu na svou rodičovskou třídu a v případě, že ji chceme získat, postačí k tomu jediný příklad a není nutné procházet XML dokument a rodičovskou třídu hledat. Relační model rovněž usnadňuje přidávání, mazání a editaci dat. Je tak snadné určit, jestli je možné smazat určitou třídu, nebo jestli jsou na ní závislé jiné elementy. Schéma DataSetu které jsem navrhl za účelem reprezentace ontologie v mé aplikaci, je v příloze A.

2.2 Popis aplikace a její schopnosti

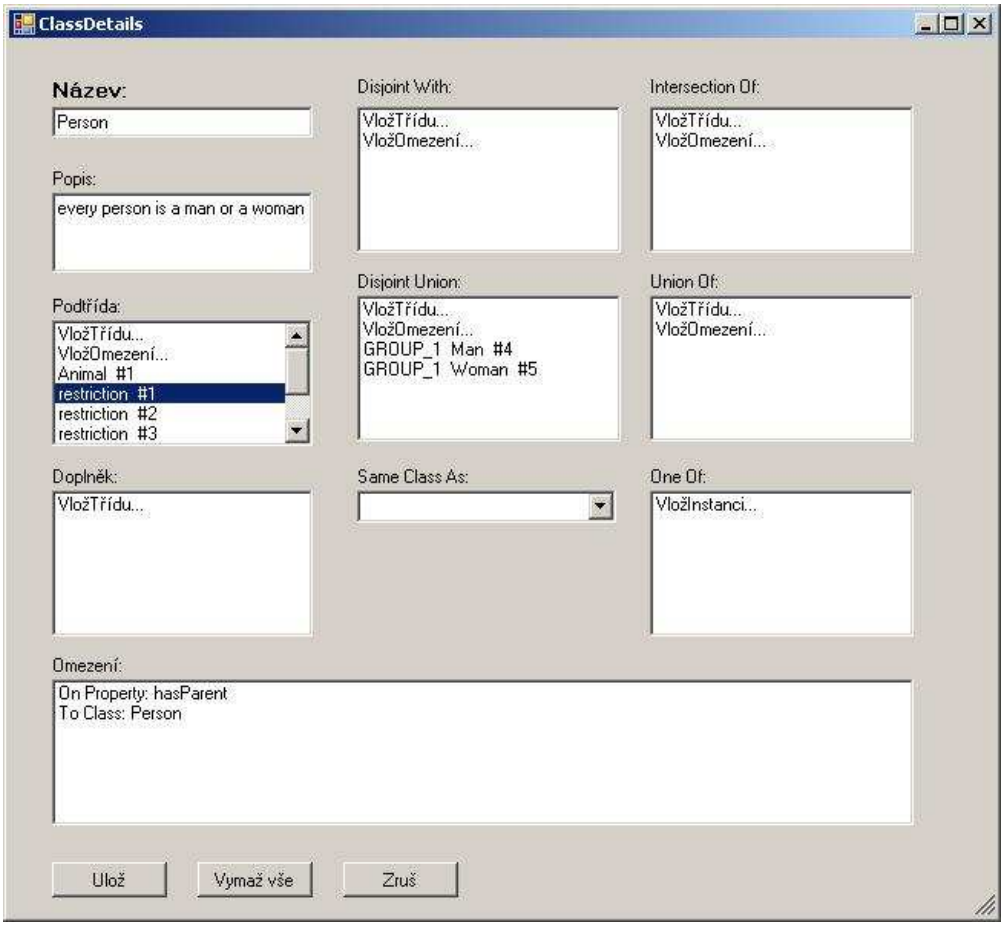
Aplikace podporuje načítání a ukládání ontologie v jazyce DAML+OIL. Rovněž umožňuje ontologii editovat a validovat. Podstatnou součástí aplikace je i rozhraní, které umožňuje ontologii testovat pomocí dotazů.

Po spuštění aplikace a načtení souboru s ontologií se zobrazí hlavní okno. V tomto okně lze přepínat mezi několika záložkami. První z nich, „Třídy“, zprostředkovává základní pohled na třídy. Pro zvolenou třídu zobrazí její hierarchii a základní údaje – jméno a popis.



Obrázek 2: Hlavní okno – třídy

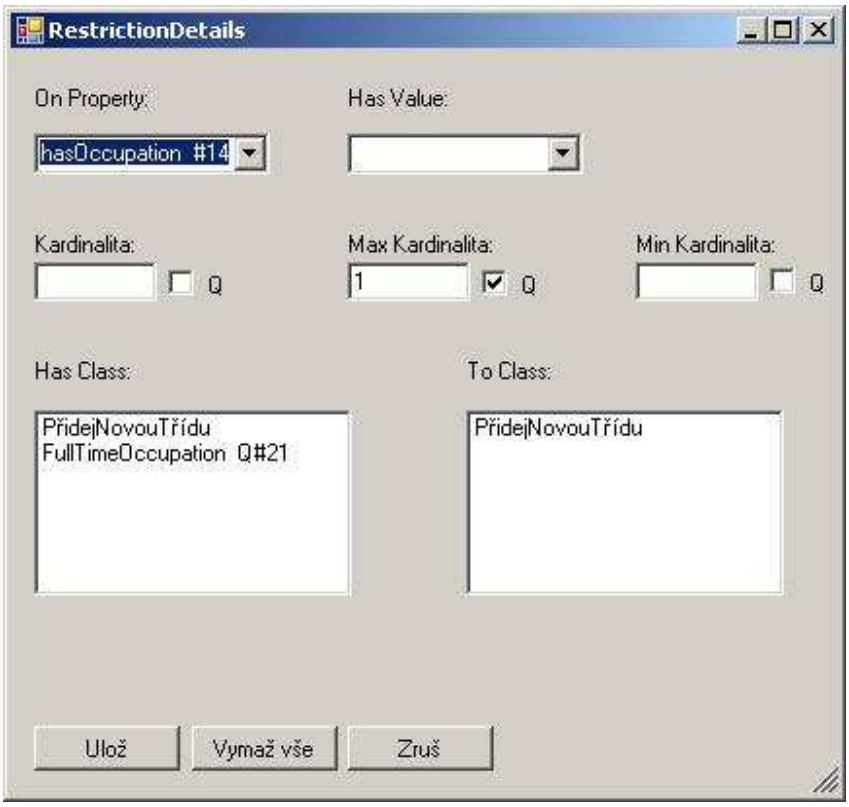
Na obrázku 2 je vidět hlavní okno „Třídy“, se zvolenou třídou „Man“. Vlevo se nachází ListBox se seznamem tříd. V tomto okně lze volit třídy a stisknutím klávesy delete lze třídu smazat. Po dvojkliku na některou z tříd se zobrazí její podrobnosti. Uprostřed okna je box se jménem třídy a další box s jejím popisem. Dole se nachází tlačítko, které umožňuje přidat novou třídu. Po jeho stisku se zobrazí podrobnosti třídy, které jsou nevyplněné. Vpravo je komponenta TreeView se zobrazenou hierarchií třídy „Man“.



Obrázek 3: Podrobnosti třídy

Na obrázku 3 je vidět okno s podrobnostmi třídy „Person“. Všechny prvky na tomto okně umožňují editaci, s výjimkou dolního ListBoxu, který zobrazuje podrobnosti omezení. V každém prvku na této stránce, který umožňuje vkládání nějakého objektu z ontologie se nachází nabídka vložení tohoto objektu. Po poklepání na položku „VložTřídu...“ se zobrazí okno s výběrem třídy. Vybraná třída se potom zobrazí v okně. Pokud poklepeme na položky „VložOmezení...“, nebo „VložInstanci...“ je situace podobná. Pokud v některém z těchto oken poklepeme na některou ze tříd, nebo omezení zobrazí

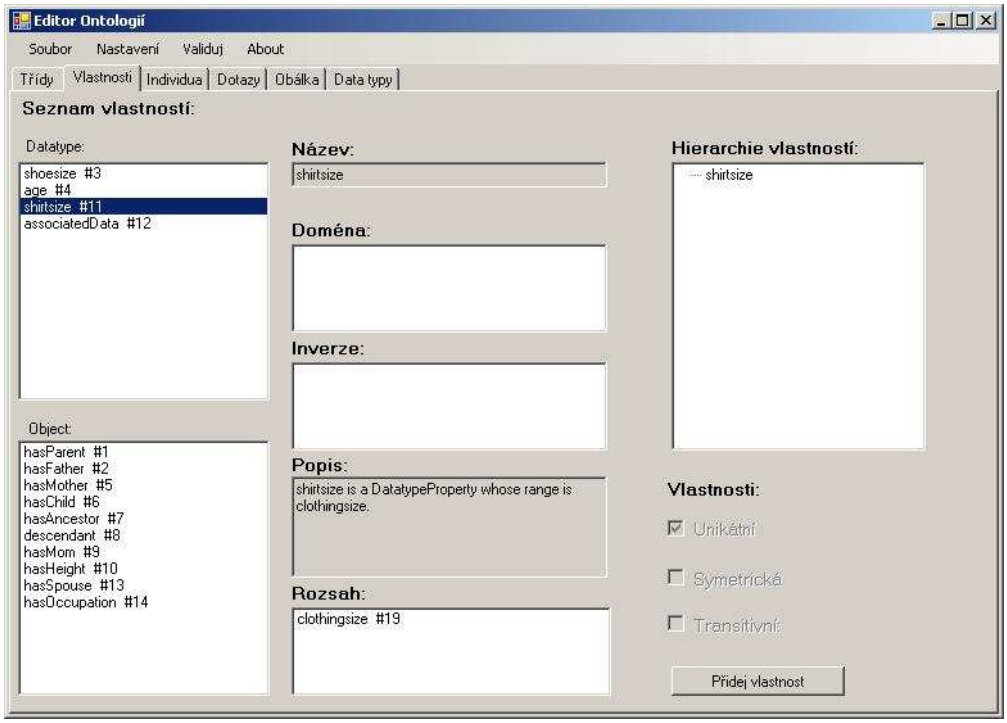
se okno s podrobnostmi o zvolené třídě, nebo omezení. Takto je možné omezení i třídy editovat.



Obrázek 4: Podrobnosti omezení

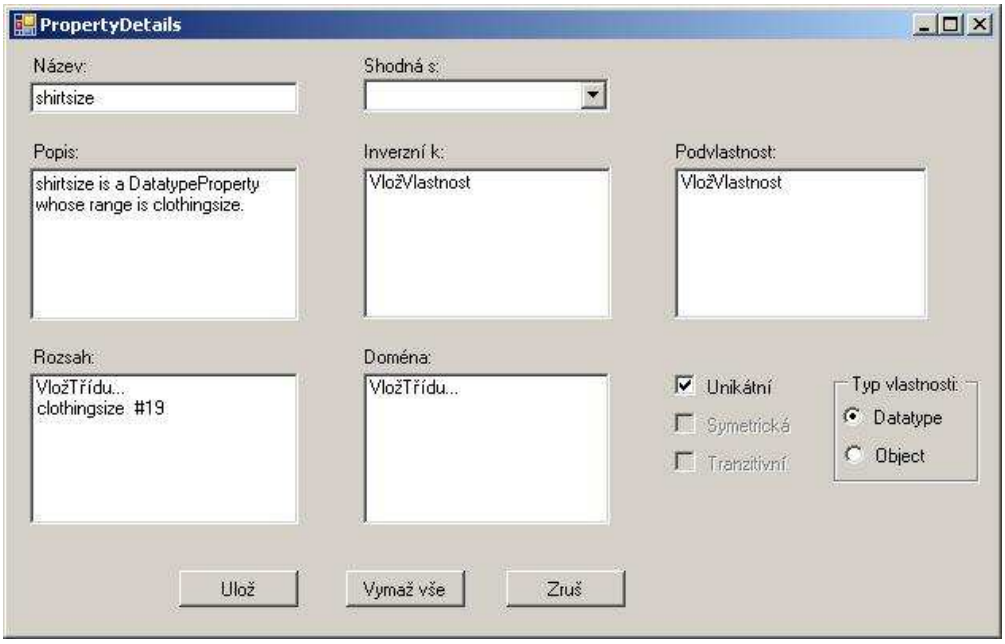
Na obrázku 4 vidíme okno s podrobnostmi omezení. Jelikož omezení nemá vlastní identifikaci (nemá jméno) a je pevnou součástí třídy, nelze omezení vytvářet samostatně, ale pouze vkládat do některého z elementů třídy (subClassOf, intersectionOf, atp.). V okně omezení, můžeme určovat vlastnost onProperty (ke které vlastnosti se omezení vztahuje), můžeme zadávat kardinalitu, eventuálně podmínky hasClass a toClass, kam můžeme po kliknutí na položku „PřidejNovouTřídu...“ vkládat třídy z patřičného seznamu. Po potvrzení okna tlačítkem „Ulož“, je omezení vloženo do odpovídající třídy.

Další ze záložek na hlavním okně, jsou „Vlastnosti“ viz. obrázek 5.



Obrázek 5: Hlavní okno – vlastnosti

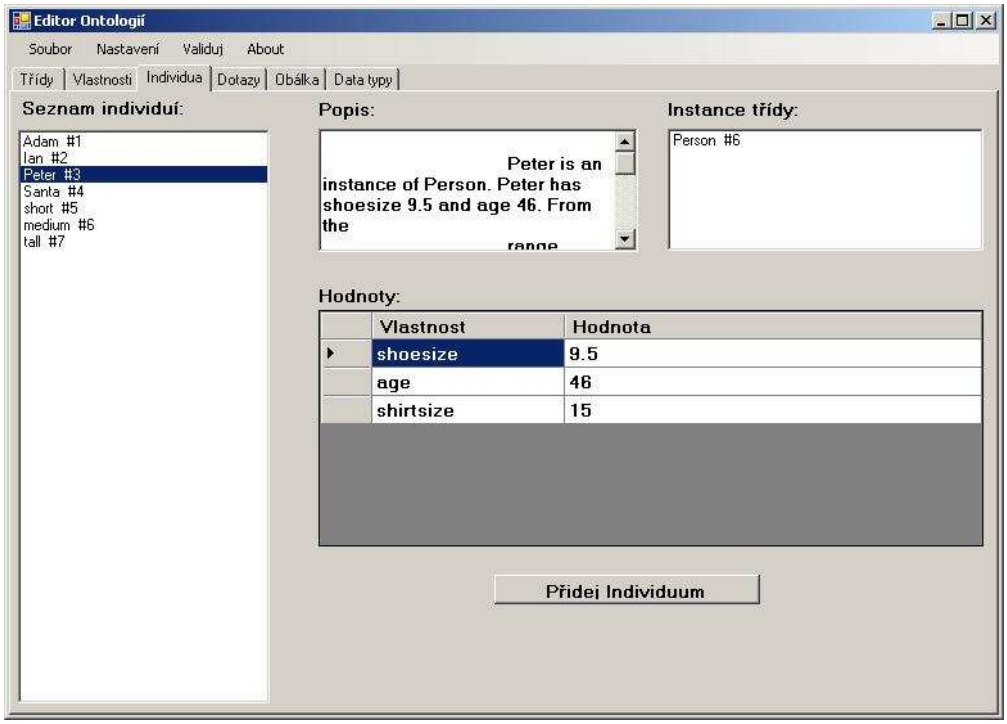
V levé části vidíme nad sebou prvky se seznamem vlastností. V prvním je seznam vlastností typu „Datatype“, ve spodním okně vidíme vlastnosti objektové. Po zvolení některé z vlastností (na obrázku je zvolena vlastnost „shirtsize“), se zobrazí její podrobnosti (název, popis, hierarchie a další). Poklepáním na některou z vlastností, se zobrazí okno s podrobnostmi dané vlastnosti, ve kterém je možné vlastnost editovat. Stejné, ale nevyplněné okno se zobrazí po stisknutí tlačítka „Přidej vlastnost“, které umožňuje tvořit nové vlastnosti. Při stisknutí tlačítka delete v některém z oken seznamu vlastností, dojde k vymazání vlastnosti.



Obrázek 6: Podrobnosti vlastnosti

Stejně jako u okna podrobnosti tříd, i zde jsou všechny prvky editovatelné. Kromě obligátního názvu a popisu, můžeme volit, zda je vlastnost shodná s jinou vlastností, zda je k nějaké vlastnosti inverzní, nebo zda je potomkem nějaké vlastnosti. Můžeme také určovat doménu (na jaké třídy lze vlastnost aplikovat) a rozsah (jakých hodnot může vlastnost nabývat). Můžeme také volit, jestli je vlastnost unikátní a v případě objektových vlastností také zda je tranzitivní, nebo symetrická. Po poklepání na název vlastnosti, nebo třídy v některém z oken, se zobrazí příslušné podrobnosti. Po poklepání na položku „VložTřídu“, nebo „VložVlastnost“ se zobrazí okno s výběrem odpovídajícího objektu.

Další ze záložek hlavního okna je záložka „Individua“. Zde můžeme vidět seznam jednotlivých individuí a jejich základní údaje (název, popis, hodnoty, mateřskou třídu).



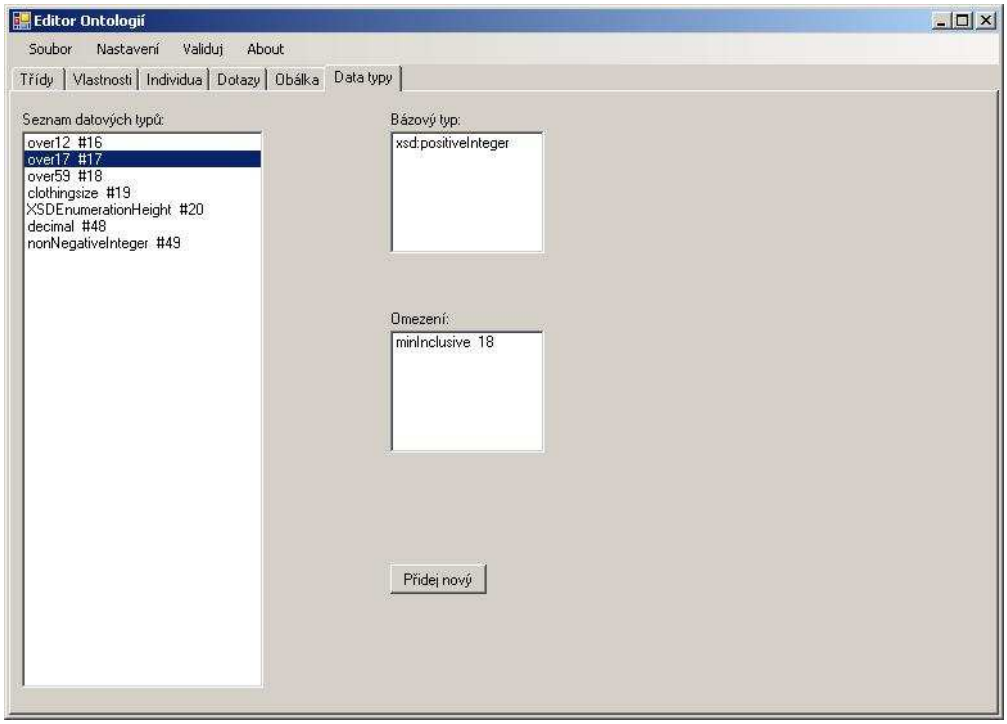
Obrázek 7: Hlavní okno – individua

Na obrázku 7 vidíme záložku „Individua“ s vybraným individuem Peter. Je to instance třídy Person a má 3 hodnoty shoesize, age a shirtsize. Po poklepání na některé z individuí, se zobrazí ono s podrobnostmi a možností editace, stejně jako po stisku tlačítka „Přidej individuum“. Stisknutím tlačítka delete, dojde k vymazání vybraného individua.

Na obrázku 8 vidíme podrobnosti individua. Lze zde editovat název, popis, lze měnit třídu, ze které je individuum odvozeno, lze určit že individuum je stejné jako jiné individuum, eventuálně že je rozdílné od jiného individua. Je zde také možné odebírat a přidávat hodnoty daného individua. Po poklepání na položku „Přidej novou hodnotu...“ se zobrazí nejprve okno s výběrem datatype vlastnosti, ze které hodnota vychází (shoesize, age ...) a poté je možné specifikovat hodnotu samotnou.

Obrázek 8: Podrobnosti individua

Aplikace podporuje také tvorbu vlastních datových typů, tzv. SimpleType. Tyto datové typy umožňují zadání bázového typu (integer, decimal, string ...) a nějakého omezení (minInclusive, maxExclusive, atp.). Např. na obrázku 9 je vidět datový typ „over17“. Ten specifikuje, že bázový typ je positiveInteger (tedy kladný celočíselný typ) a omezení minInclusive 18 (tedy rozsah 18 až nekonečno). Pomocí těchto typů a omezení se pak mohou definovat třídy jako Senior, kde je dáno že senior je osoba a zároveň má omezení vlastnosti age, na třídu over59.



Obrázek 9: Hlavní okno - data typy

Poklepáním na příslušný datový typ, nebo stiskem tlačítka „Přidej nový“ je zobrazeno okno s podrobnostmi datového typu. Stiskem delete na seznamu datových typů, je tento vymazán.

V okně podrobnosti datového typu (obrázek 10), je možné volit název, bázový typ (může jich být několik) a omezení (taktéž jich může být několik). Konkrétně na obrázku 10 jsou zobrazeny detaily typu over17.

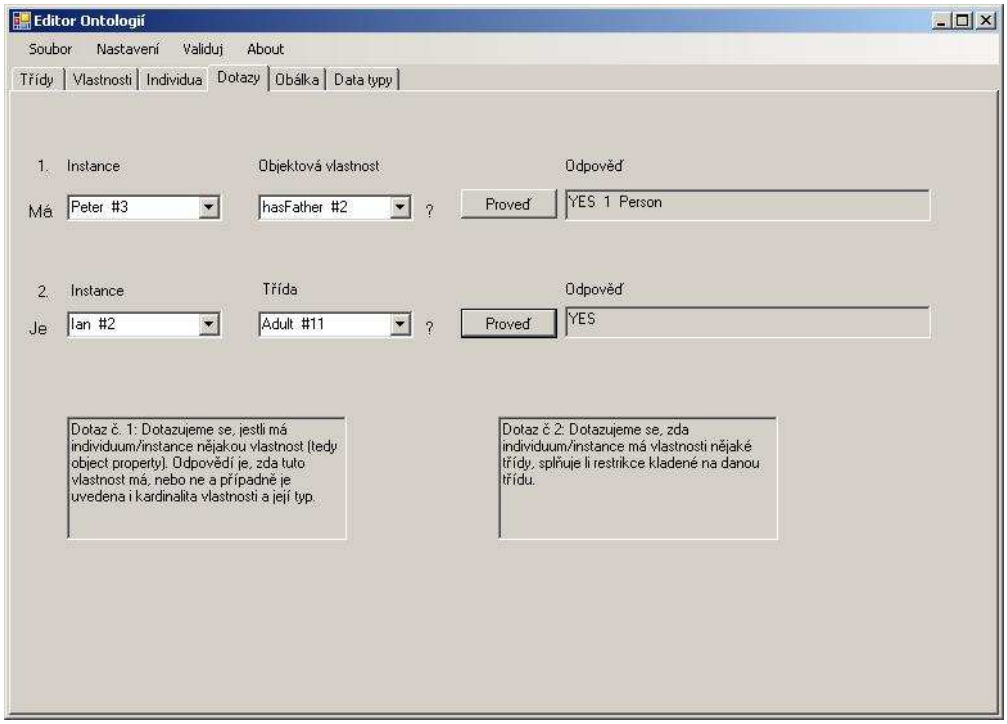
Podstatnou funkcí této aplikace, je i možnost testovat ontologie. Toto testování se děje pomocí dotazů kladených v záložce „Dotazy“ hlavního okna (obrázek 11). Jak je vidět, umožněny jsou dva typy dotazů. Prvním typem dotazu zjišťujeme, zda individuum má nějakou objektovou vlastnost. Na obrázku 11 konkrétně vidíme dotaz, zda individuum Peter má vlastnost

hasFather. V odpovědi je uvedeno že vlastnost má a to jednu, jejíž typ je třída Person.

Obrázek 10: Podrobnosti datového typu

Dalším typem dotazu je otázka, zda individuum má vlastnost nějaké třídy, resp. Jestli splňuje restrikce kladené na danou třídu. Na obrázku jedenáct je vidět dotaz, zda individuum Ian má vlastnosti třídy Adult. Podle odpovědi tyto vlastnosti má. Třída adult je v této ontologii definována jako průnik třídy Person (což Ian je) a omezení na vlastnost age, která „má třídu“ over17, což je SimpleType, definovaný jako celočíselný kladný int s hodnotou 18 a víc. Individuum Ian má hodnotu age 37, proto toto omezení splňuje.

Další ze schopností aplikace, je validátor ontologie. Jeho výstup vidíme na obrázku 12. Je zobrazeno, jestli je ontologie validní, nebo ne a ve spodním seznamu jsou zobrazena hlášení validátoru. Na obrázku vidíme, že daná ontologie není validní a že nesouhlasí předepsané kardinality hodnot u individuí Ian a Santa.



Obrázek 11: Hlavní okno - dotazy

Obrázek 12: Validátor ontologie

Další záložkou hlavního okna je „Obálka“, kde lze editovat obálku ontologie. Tedy komentář k ontologii a verzi jazyka, kterým je ontologie napsána.

Aplikace obsahuje také menu, pomocí kterého lze načítat a ukládat ontologii, spouštět validátor a nastavovat jmenné prostory.

2.3 Struktura aplikace

Aplikace se skládá ze 3 hlavních částí. Každé této části odpovídá ve Visual Studiu jeden projekt. Jsou to OntologyManagement, GUI a CommonElements.

Nejjednodušší je modul CommonElements. Jsou zde obsaženy třídy a struktury, které jsou využívány jak metodami v managementu, tak v GUI. Jedná se například o strukturu návratové hodnoty validátoru, která je definována následovně:

```
public class ValidatorOutput
    {
        public bool IsValid;
        public ArrayList Messages;

        public ValidatorOutput()
        {
            Messages = new ArrayList();
            IsValid = true;
        }
    }
```

Skládá se z příznaku, zda je ontologie validní a ze seznamu hlášení validátoru. Má také bezparametrický konstruktor, který inicializuje příslušné proměnné. Nejdůležitější součástí projektu CommonElements, je však DataSet reprezentující ontologii. Jeho schéma jsem vzhledem k jeho rozsahu umístil do přílohy A.

Projekt GUI je jak zkratka napovídá grafické uživatelské rozhraní. Jsou zde nadefinovány všechny formuláře a jejich logika (funkce tlačítek, naplnění ovládácích prvků atp.).

Nejpodstatnější částí aplikace z hlediska funkčnosti je projekt OntologyManagement. Jsou zde implementovány dvě klíčové třídy. DataSetHandler a OntologyValidator. Jak název třídy DataSetHandler napovídá, slouží k provádění všech operací nad DataSetem s ontologií. Stará se o jeho naplnění a editaci, je zde také implementována logika dotazů.

Třída OntologyValidator slouží k validování ontologie. Hlavní metoda Validate, postupně validuje celou ontologii. Návratovou hodnotou je výše popsaná třída ValidatorOutput. Validace probíhá v několika krocích. Nejprve se provede kontrola duplicitních hodnot. Tedy, jestli například některá třída nemá dva elementy subClassOf nastaveny na stejnou třídu. Dále se zkontrolují protiřečící si hodnoty. Tedy jestli např. některá třída nemá uvedenu stejnou třídu v tabulce subClassOf a complementOf. Tyto testy se provádí pro třídy, vlastnosti i individua. Pro individua je navíc provedena i kontrola jejich hodnot. Kontroluje se, zda odpovídá hodnota datovému typu (tedy jestli není předepsán int a hodnota není třeba „ahoj“). Kontroluje se kardinalita hodnot (například vlastnost age je unique, proto každé individuum musí mít maximálně jednu hodnotu typu age). Kontroluje se také, jestli v případě definování domain a range u vlastností, splňují odpovídající hodnoty toto omezení. V případě nalezení chyby, je příznak IsValid v návratové třídě nastaven na *false* a do kolekce Messages je přidána patřičná zpráva. Výsledky validace jsou pak zobrazeny uživateli v grafickém rozhraní.

Do tohoto textu jsem neumístil výpisy zdrojového kódu, všechen kód je dostupný na přiloženém DVD i s komentáři.

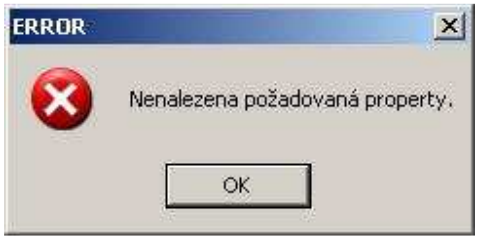
2.4 Použití aplikace

Tato kapitola, by měla poskytnout stručný návod na použití aplikace. Po jejím přečtení by měl být uživatel schopen s programem pracovat.

K přístupu k jednotlivým základním funkcím aplikace slouží menu. To má následující základní položky: Soubor, Nastavení, Validuj a About. V položce soubor najdeme klasickou paletu funkcí: Načtení a uložení souboru, založení nového souboru a ukončení aplikace. Po zvolení nabídky uložit, je ontologie z DataSetu uložena do souboru ve formátu DAML+OIL. Naopak při načtení souboru, je DataSet zinicializován z DAML+OIL souboru. V případě, že načítaný soubor není ve formátu podporovaným aplikací, zobrazí se chybová hláška.

Obrázek 13: Selhání načtení souboru

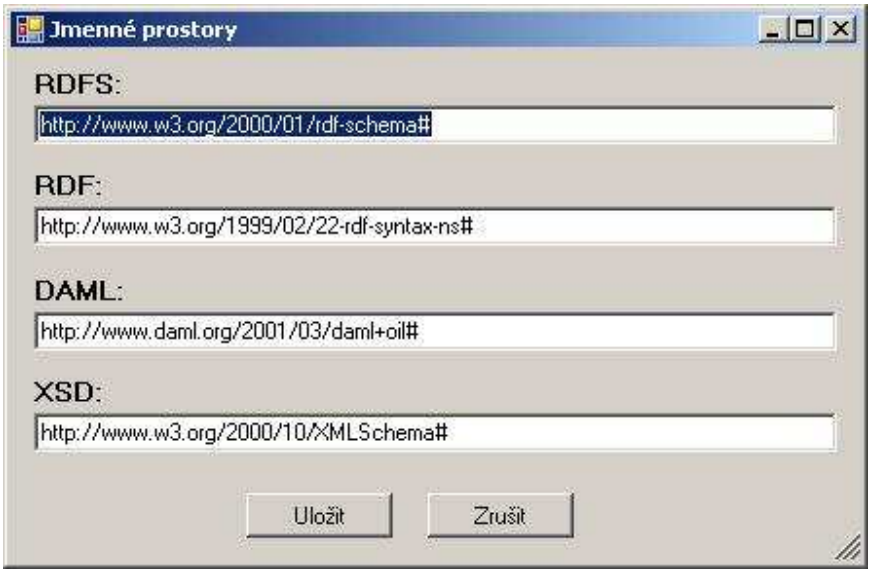
Pokud je soubor v čitelném formátu, ale je v něm nějaká syntaktická chyba, zobrazí se hláška s popisem dané chyby:



Obrázek 14: Načtení souboru - syntaktická chyba

Po úspěšném načtení souboru je zinicializován DataSet a vyplněny prvky na formuláři. Při zvolení položky „Zavřít“ je zobrazen dotaz a po jeho potvrzení je aplikace ukončena.

Další položkou menu je „Nastavení“. Zde se nachází jediná položka – „Jmenné prostory“. Po jejím vybrání se zobrazí okno s nastavením jmenných prostorů.



Obrázek 15: Nastavení jmenných prostorů

Další položkou hlavního menu je „Validuj“. Po kliknutí se zobrazí okno validátoru (viz. obr. 12). Zde se nachází dvě tlačítka. Po stisku tlačítka Validuj, je zobrazena zpráva o validitě ontologie. Tlačítko zavřít zavře okno validátoru.

V poslední položce menu „About“, se skrývá podnabídka „O aplikaci“, která zobrazí okno s informací o názvu aplikace a autorovi.

Práce s aplikací je jednoduchá. V záložkách hlavního okna „Třídy“, „Vlastnosti“, „Individua“ a „Datové typy“ jsou podporovány stejné operace. Po kliknutí na položku seznamu (v případě vlastností dvou seznamů) v levé části okna, se zobrazí informace o daném objektu. Po stisknutí tlačítka delete v tomto

okně, dojde k vymazání objektu. To se však provede pouze v případě, že na něm nezávisejí jiné podřízené objekty. Před smazáním je ještě po uživateli žádáno potvrzení. Dvojklik na některou z položek otevře okno s detaily daného objektu, kde je možné objekt editovat. Stejné okno, ovšem bez vyplněných údajů se otevře i při stisku tlačítka přidání nového objektu (třídy, individua, typu, vlastnosti). Obrázky těchto oken viz výše.

V okně s podrobnostmi třídy, je možné editovat název a popis třídy. Dále je zde možné editovanou třídu definovat pomocí vkládání dalších objektů do příslušných polí. Každé z polí odpovídá nějaké konstrukci jazyka DAML+OIL (podtřídy, průniky, sjednocení ...). V každém poli nahoře, je položka, která umožňuje přidání objektu který se do daného pole hodí (třída, omezení individuum). Pro přidání objektu stačí dvojkliknout na odpovídající položku. V případě omezení, se jeho podrobnosti zobrazí po najetí na položku omezení. K zobrazení podrobností o omezení, slouží pole úplně dole na formuláři. V případě dvojkliku na třídu nebo omezení v některém z polí, se zobrazí okno s podrobnostmi. V případě třídy se jedná o totéž okno. Po stisku delete na příslušném okně, dojde k odstranění objektu z definice dané třídy. Objekt jako takový však v ontologii zůstává (netýká se omezení, a virtuálních tříd).

V okně s podrobnostmi omezení musíme pomocí ComboBoxu zvolit vlastnost, ke které se omezení vztahuje. Pak můžeme přidávat další podmínky. Každý z prvků na formuláři opět odpovídá konstrukcím jazyka DAML+OIL pro omezení.

Dalším oknem umožňujícím editaci jsou podrobnosti vlastnosti. Zde můžeme pomocí RadioButtonu zvolit, jedná-li se o datatypovou, nebo objektovou vlastnost. Podle toho se upraví možnosti voleb transitivity a symetričnosti. Další pole fungují stejně jako v případě tříd.

Stejně jako předchozí editační okna, funguje i okno podrobností individua. A datového typu. Je možné přidávat objekty dvojkliknutím na odpovídající položku a je možné objekty mazat stiskem klávesy delete.

V záložce „Obálka“ hlavního okna, je možné měnit odpovídající vlastnosti ontologie (popis a verze ontologického jazyka) prostým vyplněním políček a stiskem tlačítka Uložit.

V záložce „Dotazy“ je pro každý dotaz nutné vybrat pomocí ComboBoxů jednotlivé objekty, na které se ptáme. V případě, že nebude některý objekt zvolen, objeví se chybová hláška.

3. Závěr

Ačkoliv je sémantický web stále zatím spíše předmětem akademického výzkumu, přibývá lidí kteří si tohoto fenoménu všímají a začínají v oblasti ontologií experimentovat. Právě pro tyto účely je určena moje aplikace. Umožňuje tvoření vlastních ontologií od základu, nebo načítání a úpravu již existujících. S těmito ontologiemi je pak možné experimentovat pomocí dotazů, nebo je validovat. Je samozřejmě možné ontologie ukládat ve formátu DAML+OIL.

Podobné aplikace jako tato, dláždí cestu k sémantickému webu začínajícím uživatelům a umožňují tak rozšiřování komunitu lidí, kteří se o tuto technologii zajímají. Časem tak dojde ke zlomení základního problému, který zatím brání většímu rozšíření sémantického webu a který v počátcích trápil i web klasický: Proč publikovat stránky s hypertextovými odkazy, když není na co odkazovat. Tento problém padl s rozšiřujícím se počtem uživatelů a je pravděpodobné, že podobný osud nakonec potká i web sémantický.

3.1 Závěr – aplikace

Možnosti aplikace byly navrženy tak, aby bylo splněno zadání této diplomové práce. Vzhledem k vyjadřovací síle jazyka DAML+OIL a času danému na vypracování práce, není v silách jediného člověka vše 100% důkladně otestovat. Proto se může stát, že uživatel narazí na nějaké chyby, nebo opomenutí, za které se jako autor předem omlouvám.

Citace

[1] Berners-Lee, T. – Hendler, J. – Lassila O. *The Semantic Web.* [online]. [cit. 2007-05-12]. URL: <http://www.sciam.com/article.cfm?articleID=00048144-10D2-1C70-84A9809EC588EF21&pageNumber=1&catID=2>

[2] Tauberer, J. *What Is RDF.* [online]. [cit. 2007-05-12]. URL: <http://www.xml.com/pub/a/2001/01/24/rdf.html>

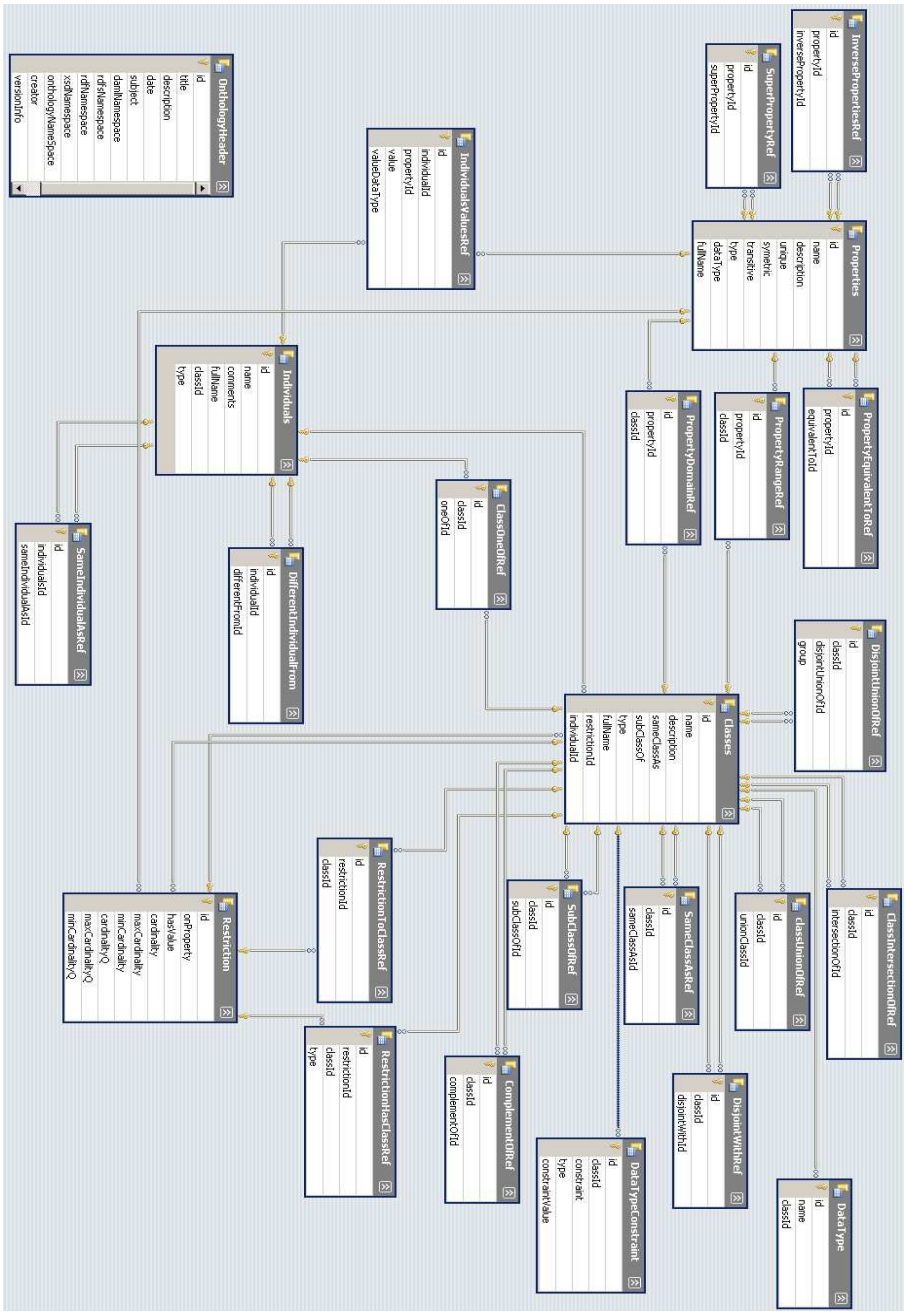
[3] Svátek, V. *Ontologie a WWW.* [online]. [cit. 2007-05-12]. URL: <http://nb.vse.cz/~svatek/onto-www.pdf>

[4] Palmer, S., B. *The Semantic Web: An Introduction.* [online]. [cit. 2007-05-12]. URL: <infomesh.net/2001/swintro>

[5] Herman, I. *Semantic Web.* [online]. [cit. 2007-05-12]. URL: <http://www.w3.org/2001/sw/>

Přílohy:

A. Schéma DataSetu pro reprezentaci ontologie



B. DVD – ROM

Součástí této práce je také disk DVD. Je na něm uložen tento text ve formátu PDF a DOC, zdrojové kódy aplikace, přeložená spustitelná aplikace, experimentální ontologie ve formátu DAML+OIL, MS .NET Framework 2.0 nutný pro běh programu a MS Visual C# 2005 Express Edition – vývojové prostředí zdarma dostupné na stránkách Microsoftu, použité pro tvorbu aplikace a vhodné pro prohlížení zdrojových kódů.